Soit un système avec des processus (threads) et des ressources à un instant donné S avec 4 structures de données :

E : nombre de ressources maximum disponible (ne change pas durant l'exécution) :

R1	R2	R3	R4
4	2	3	1

A : nombre de ressources disponibles à l'instant S de l'exécution :

R1	R2	R3	R4
2	1	0	0

Alloc : tableau de l'allocation de chaque processus en fonction des ressources disponibles à l'instant S de l'exécution:

	R1	R2	R3	R4
P1	0	0	1	0
P2	2	0	0	1
P3	0	1	2	0

Req : tableau des ressources qui seront demandées par chaque processus dans le futur i.e. après S:

	R1	R2	R3	R4
P1	2	0	0	1
P2	1	0	1	0
P3	2	1	0	0

S est-il sur ?

Rép. On va appliquer l'algorithme du banquier :

Itération no 1 :

- Est-ce que l'ordonnanceur peut alloué à P1 les ressources demandés après l'instant S ?

	R1	R2	R3	R4
A :	2	1	0	0
Req(P1) :	2	0	0	1

On voit qu'au niveau de R4 il n'y a pas assez de ressources disponibles à l'instant S, donc la réponse est non (n.b. ça prend au moins 1 des 4 quatre qui demande plus de ressources que disponible à l'instant S pour dire non).

- Est-ce que l'ordonnanceur peut alloué à P2 les ressources demandés après l'instant S ?

	R1	R2	R3	R4
A :	2	1	0	0
Req(P2) :	1	0	1	0

On voit qu'au niveau de R3 il n'y a pas assez de ressources disponibles à l'instant S, donc la réponse est non.

- Est-ce que l'ordonnanceur peut alloué à P3 les ressources demandés après l'instant S ?

	R1	R2	R3	R4
A :	2	1	0	0
Req(P3):	2	1	0	0

On voit que les ressources demandés par P3 sont suffisantes, donc oui

- Mise à jour de A quand P3 aurait terminé avant d'aller à la prochaine itération :

Ça prend au moins un oui pour passer à l'itération suivante, ce qui est le cas. Ça veut donc dire qu'on suppose que l'ordonnanceur va allouer les ressources à P3 et lui donner la main pour qu'il s'exécute jusqu'à la fin c'est-à-dire de manière à libérer les ressources acquises dans Req(P3) mais aussi les ressources demandées disponibles dans A. Par conséquent, à la fin de son exécution, P3 va donc libérer les ressources Alloc(P3) et celle acquises dans A via Req(P3). À la fin on peut dire que la nouvelle valeur de A nommons la A' sera égale à A + Alloc(P3) donc :

Alloc(P3) :	0	1	2	0
A :	2	1	0	0
A' :	**2**	**2**	**2**	**0**

Itération no 2 :

- D'abord A' devient A pour cette nouvelle itération et on se concentre maintenant sur P1 et P2.

- Est-ce que l'ordonnanceur peut alloué à P1 les ressources demandés après l'instant S ?

	R1	R2	R3	R4
A :	2	2	2	0
P1 :	2	0	0	1

On voit qu'au niveau de R4 même si P3 est exécuté et terminé, il n'y a pas assez de ressources, donc la réponse est non.

- Est-ce que l'ordonnanceur peut alloué à P2 les ressources demandés après l'instant S ?

	R1	R2	R3	R4
A :	2	2	2	0
P2 :	1	0	1	0

On voit que les ressources demandés par P2 sont suffisantes, donc oui

- Mise à jour de A quand P2 aurait terminé avant d'aller à la prochaine itération :

Ça veut donc dire qu'on suppose que l'ordonnanceur va allouer les ressources à P3 et lui donner la main pour qu'il s'exécute jusqu'à la fin c'est-à-dire de manière à libérer les ressources acquises dans Req(P2) mais aussi les ressources demandées disponibles dans A. Par conséquent, à la fin de son exécution, P2 va donc libérer les ressources Alloc(P2) et celle acquises dans A via Req(P2). À la fin on peut dire que la nouvelle valeur de A nommons la A' sera égale à A + Alloc(P2) donc :

Alloc(P2) :	2	0	0	1
A :	2	2	2	0
A' :	**4**	**2**	**2**	**1**

Itération no 3 :

- D'abord A' devient A pour cette nouvelle itération et on se concentre maintenant sur P1.

- Est-ce que l'ordonnanceur peut alloué à P1 les ressources demandés après l'instant S ?

	R1	R2	R3	R4
A :	4	2	2	1
P1 :	2	0	0	1

On voit que les ressources demandés par P1 sont suffisantes, donc oui

- Mise à jour de A quand P1 aurait terminé :

Ça veut donc dire qu'on suppose que l'ordonnanceur va allouer les ressources à P3 et lui donner la main pour qu'il s'exécute jusqu'à la fin c'est-à-dire de manière à libérer les ressources acquises dans Req(P1) mais aussi les ressources demandées disponibles dans A. Par conséquent, à la fin de son exécutio, P1 va donc libérer les ressources Alloc(P1) et celle acquises dans A via Req(P1). À la fin on peut dire que la nouvelle valeur de A nommons la A' sera égale à A + Alloc(P1) donc :

Alloc(P1) :	0	0	1	0
A :	4	2	2	1
A' :	**4**	**2**	**3**	**1**

On voit donc que A' final = E, on a donc récupérer les ressources.

En conclusion, ça veut dire que si on est dans l'état S, on est pas dans une situation d'interblocage puisque P3 pourrait d'abord s'exécuter, puis P2, puis finalement P1. C'est ce qu'on nomme un état sur.

Maintenant, on supposons qu'à l'état S on reçoit la demande R1 de P1, est-ce que l'OS doit accorder la ressource ?

Rép. Regardons l'état S' successeur de S après avoir alloué R1 à P1 :

E' : nombre de ressources maximum disponible (ne change pas durant l'exécution) :

R1	R2	R3	R4
4	2	3	1

A' : nombre de ressources disponibles à l'instant S de l'exécution :

R1	R2	R3	R4
1	1	0	0

Alloc' : tableau de l'allocation de chaque processus en fonction des ressources disponibles à l'instant S de l'exécution:

	R1	R2	R3	R4
P1	1	0	1	0
P2	2	0	0	1
P3	0	1	2	0

Req' : tableau des ressources qui seront demandées par chaque processus dans le futur i.e. après S:

	R1	R2	R3	R4
P1	1	0	0	1
P2	1	0	1	0
P3	2	1	0	0

Est-ce que l'ordonnanceur peut alloué à P1 ou P2 ou P3 les ressources demandés après l'instant S'?

	R1	R2	R3	R4	
A' :	1	2	0	0	
Req'(P1) :	1	0	0	1	non
Req'(P2) :	1	0	1	0	non
Req'(P3) :	2	1	0	0	non

S' est non sur donc la demande de R1 par P1 conduit à un état non sûr et devrait être rejetée par le système.